1 Neural Network

Neural network is a powerful machine learning model for classification and regression. It has many architectures and learning algorithms. We will introduce representative ones.

1.1 Neuron

Generally speaking, a neuron is a graph representation of some nonlinearized linear function. Let \( f(x) = x^T \beta \) be a linear function, and \( \sigma : \mathbb{R} \to \mathbb{R} \) be some (nonlinear) function.

Graphically, a neuron is a node taking weighted sum of elements in \( x \) as input, each \( x_j \) being weighted by \( \beta_j \), and outputting the sum nonlinearized by \( \sigma \).

[Demo] Draw structure of a neuron.

[Remark] A neuron takes different features of one instance as input.

Element \( \beta_j \) is called weight and \( \sigma \) is called activation function. There are different types of activation functions; a common one is logistic sigmoid function

\[
\sigma(z) = \frac{1}{1 + \exp(-z)}. \tag{1}
\]

For more activation functions, see e.g. https://en.wikipedia.org/wiki/Activation_function

1.2 Feed-Forward Neural Network

Roughly speaking, a neural network is a network of neurons e.g. Figure 1.

![Neural Network Architecture](image)

Figure 1: Neural Network Architecture [PRML, Figure 5.1]

The network has many connected layers. An instance is input from input layer and travels through hidden layers; its prediction is output at output layer.

A layer contains a set of neurons, each taking neuron outputs from previous layer as input, and sending its output to next layer. Typically, neurons in one layer are independent and have the same activation function.
For classification, one typically constructs one neuron for each class at output layer e.g. to classify cat and dog in image, we can construct one output neuron indicating whether the image contains a cat or not, and another output neuron indicating if the image contains dog or not.

Weights are the unknown parameters of a neural network. As more neurons are added to one layer, and more layers are added to the network, we will have more weights thus a bigger model (which is more likely to overfit). Training neural network is challenging.

[Exercise] Count the number of weights in a neural network.

[Discussion] What happens if all activation functions are linear?

### 1.3 Learning Neural Network by Back-Propagation

Suppose there are $n$ training instances. The loss function at one output neuron $\sigma$ is

$$L_n = \sum_{i=1}^{n} (\sigma(x_i) - y_i)^2.$$  

(2)

We want to learn a neural network i.e. a set of weights $w$'s that minimizes $L_n$.

It is clear that critical point method is not suitable, since function $\sigma$ can be highly complex (thus getting a closed-form solution is infeasible). So we will try numeric optimization techniques.

Let's try gradient descent. This means we should evaluate gradient of $L_n$ w.r.t. each weight in the network. We will need to apply the chain rule.

[Exercise] Let's derive the gradient w.r.t. two weights in a small neural network.

Once we have gradients e.g.

$$\frac{\partial}{\partial w} L_n = \sum_{i=1}^{n} 2[\sigma(x_i) - y_i] \cdot \sigma'(x_i).$$  

(3)

We can compute each term using input training instances and update gradients, and update

$$w = w - \eta \cdot \frac{\partial}{\partial w} L_n.$$  

(4)

So learning a neural network can be summarized in two steps: (1) fix network and input training instances to get predictions; (2) based on predictions, compute gradients and update network weights. This style of learning algorithm is called back-propagation.

One issue is using all training instances to update gradient can be expensive. In practice we can compute gradient using a single instance at each update iteration i.e.

$$\frac{\partial}{\partial w} L_n = 2[\sigma(x_i) - y_i] \cdot \sigma'(x_i).$$  

(5)

This technique is called stochastic gradient descent. We will defer it to online learning.

Another issue is we can avoid overfitting by adding a regularization term to loss function i.e.

$$L_n = \sum_{i=1}^{n} (\sigma(x_i) - y_i)^2 + \lambda \sum_{w \in \text{ANN}} w^2,$$  

(6)
where \( w \in ANN \) means any weight in the network. This technique is called weight decay.

[Exercise] What would the new gradient look like?

Another approach to avoid overfitting is early stop i.e. we stop gradient update before some standard convergence criterion is met.

[Reading] ELS Chapter 11.3, 11.5 (11.5.1, 11.5.2, 11.5.4.)

[Other Reference] PRML, Chapter 5.